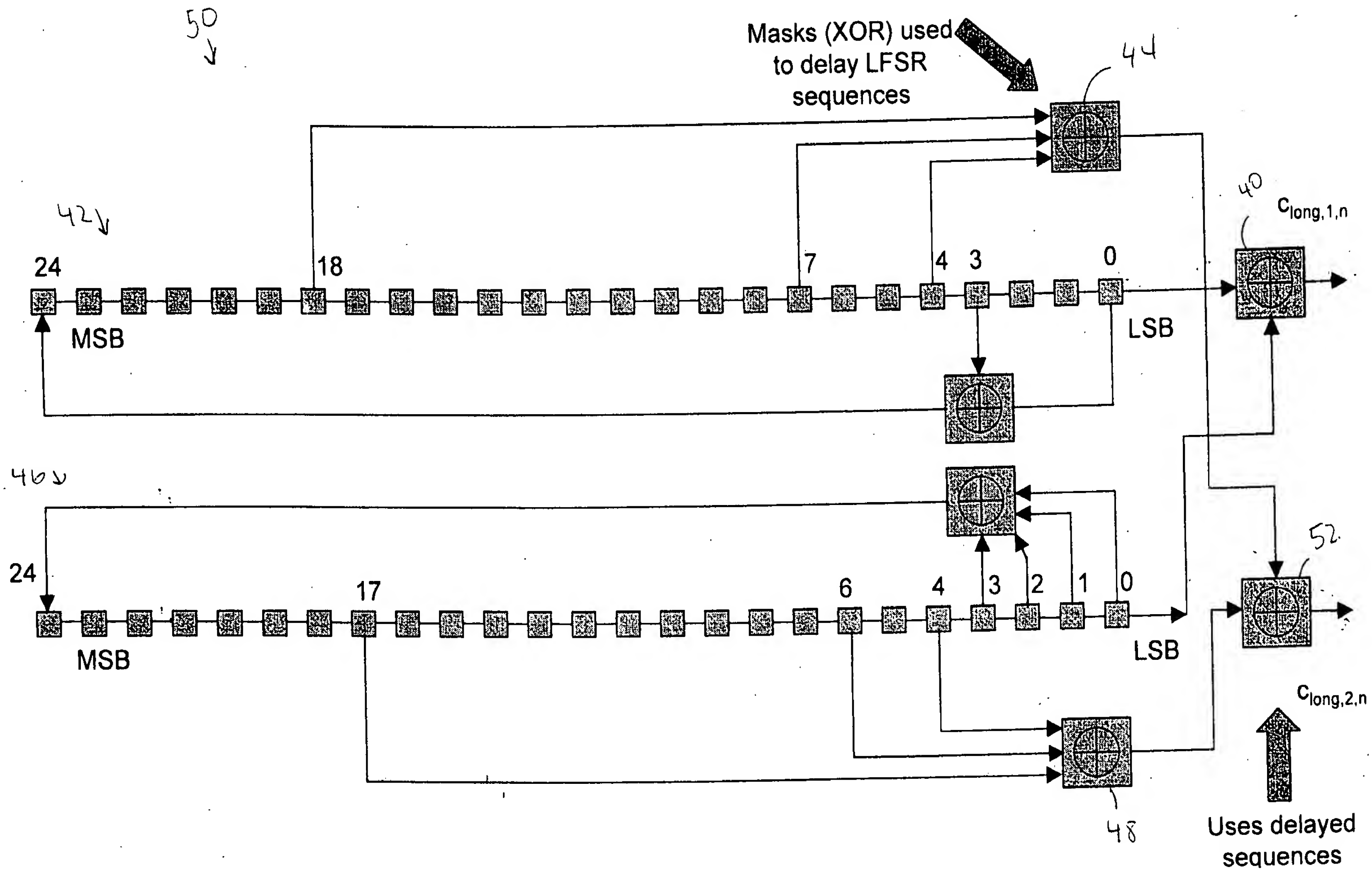
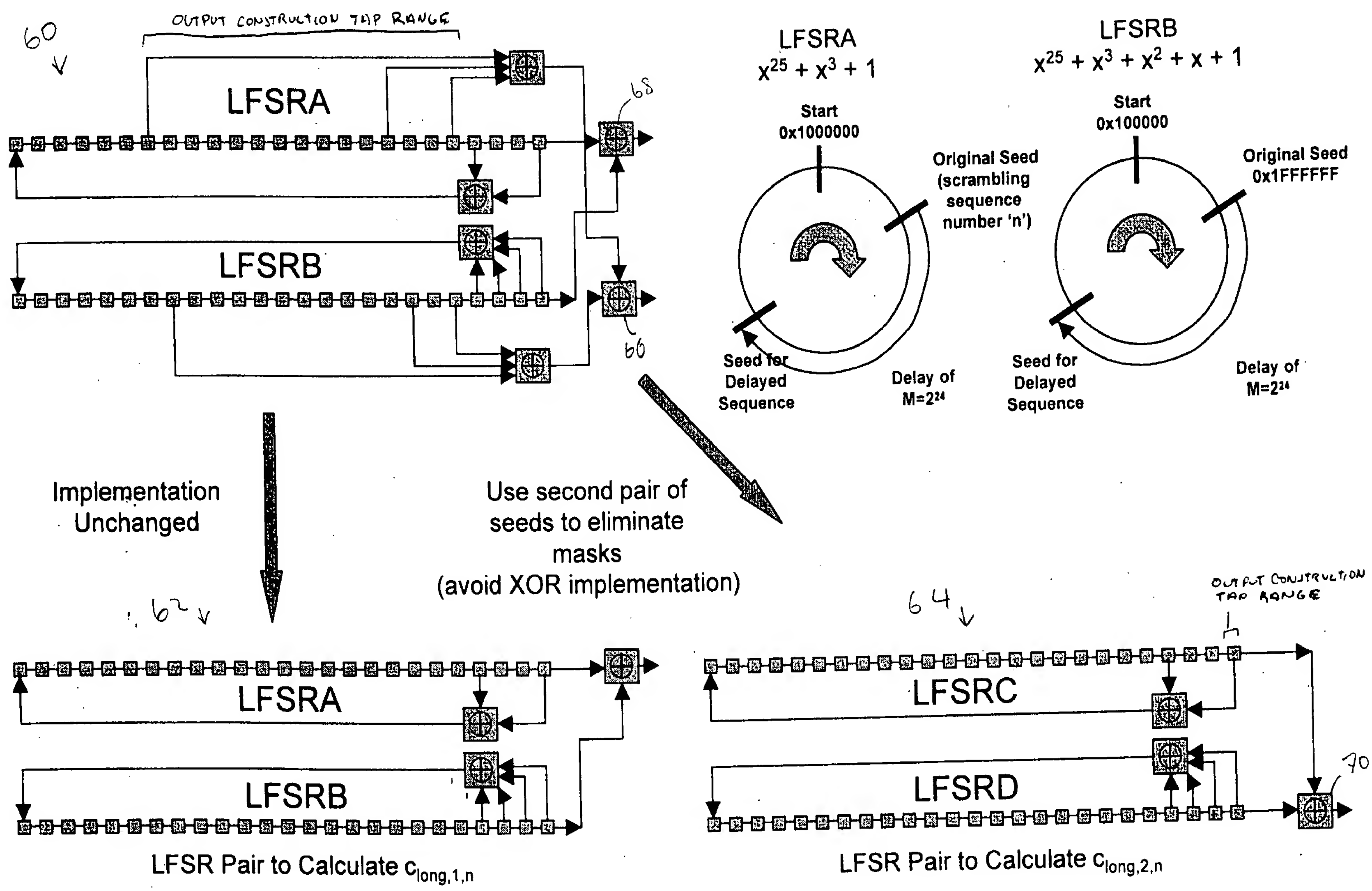


FIGURE 1
(PRIOR ART)



60
OUTPUT CONSTRUCTION TAP RANGE
LFSRA
LFSRB
68
66
LFSRA
$x^{25} + x^3 + 1$
Start
0x1000000
Original Seed
(scrambling
sequence
number 'n')
Seed for
Delayed
Sequence
Delay of
$M=2^{24}$
LFSRB
$x^{25} + x^3 + x^2 + x + 1$
Start
0x100000
Original Seed
0x1FFFFFF
Seed for
Delayed
Sequence
Delay of
$M=2^{24}$
Implementation
Unchanged
Use second pair of
seeds to eliminate
masks
(avoid XOR implementation)
62
LFSRA
LFSRB
LFSR Pair to Calculate $c_{long,1,n}$
64
OUTPUT CONSTRUCTION
TAP RANGE
LFSRC
LFSRD
70
LFSR Pair to Calculate $c_{long,2,n}$

FIGURE 2

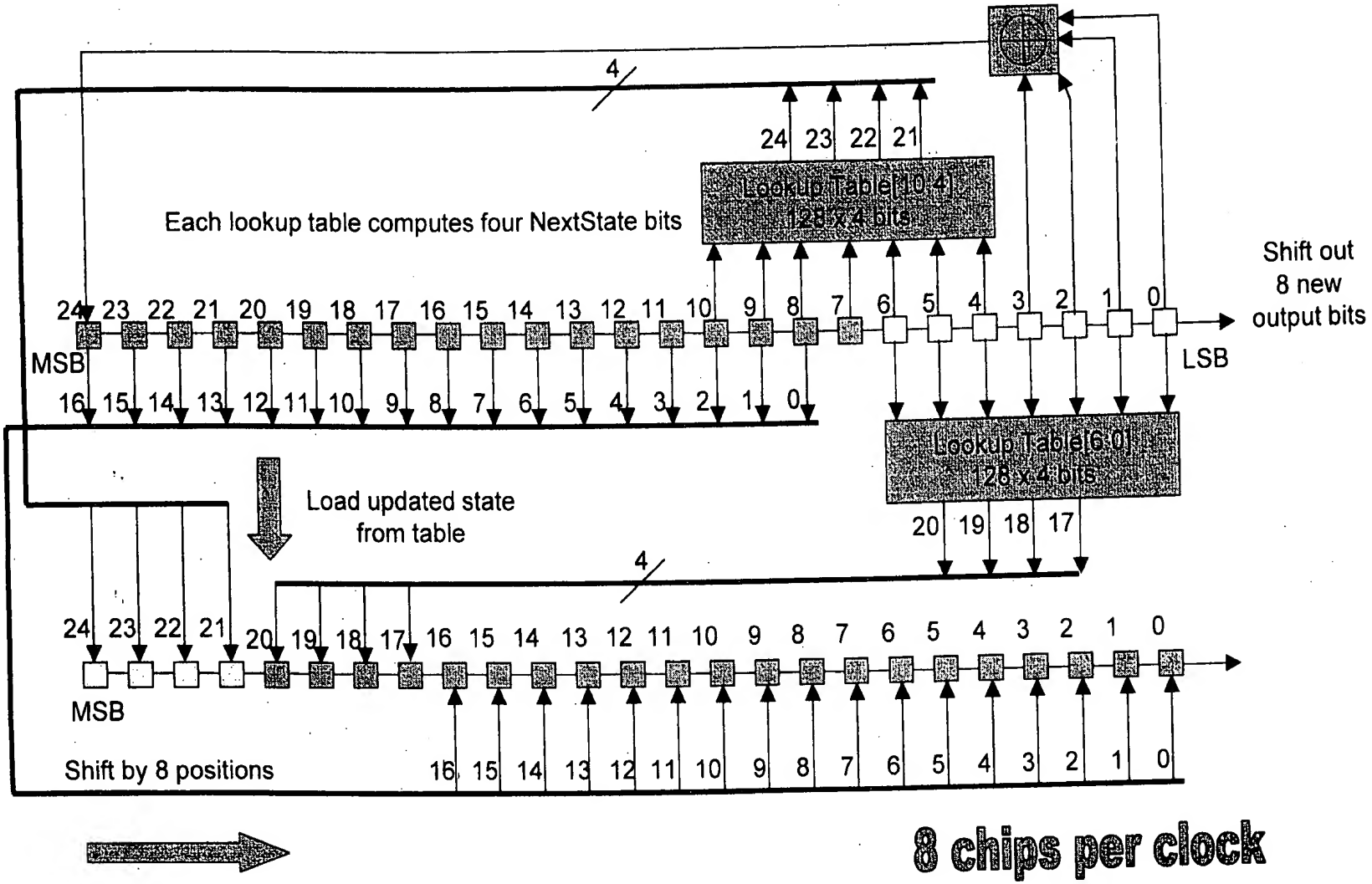

4
24 23 22 21
Lookup Table[10:4]
128 x 4 bits
Each lookup table computes four NextState bits
Shift out
8 new
output bits
24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
MSB
LSB
16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Lookup Table[6:0]
128 x 4 bits
Load updated state
from table
20 19 18 17
4
24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
MSB
Shift by 8 positions
16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
8 chips per clock

FIGURE 3

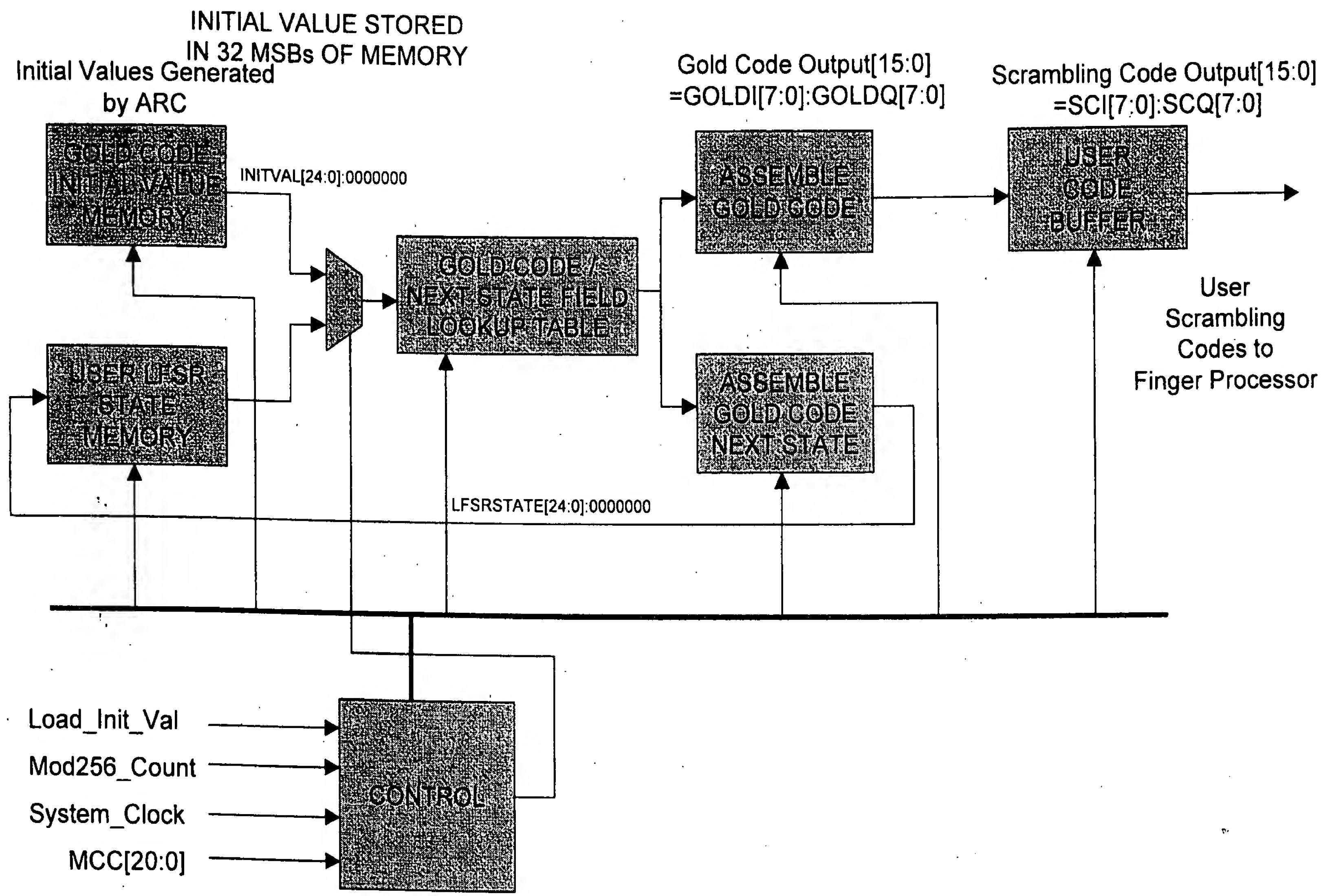

INITIAL VALUE STORED
IN 32 MSBs OF MEMORY
Initial Values Generated
by ARC
GOLD CODE
INITIAL VALUE
MEMORY
INITVAL[24:0]:0000000
USER LFSR
STATE
MEMORY
GOLD CODE /
NEXT STATE FIELD
LOOKUP TABLE
Gold Code Output[15:0]
=GOLDI[7:0]:GOLDQ[7:0]
ASSEMBLE
GOLD CODE
ASSEMBLE
GOLD CODE
NEXT STATE
LFSRSTATE[24:0]:0000000
Scrambling Code Output[15:0]
=SCI[7:0]:SCQ[7:0]
USER
CODE
BUFFER
User
Scrambling
Codes to
Finger Processor
Load_Init_Val
Mod256_Count
System_Clock
MCC[20:0]
CONTROL

FIGURE 4

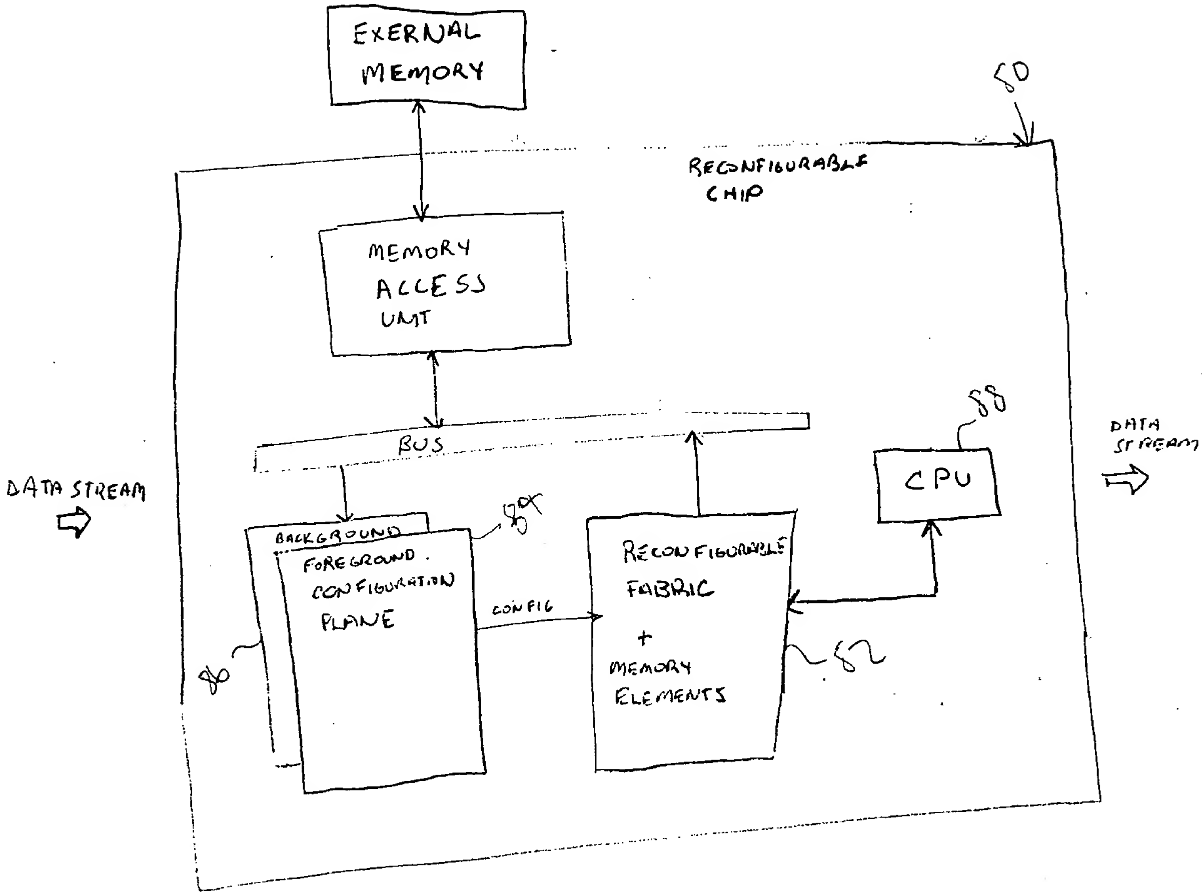


FIGURE 5

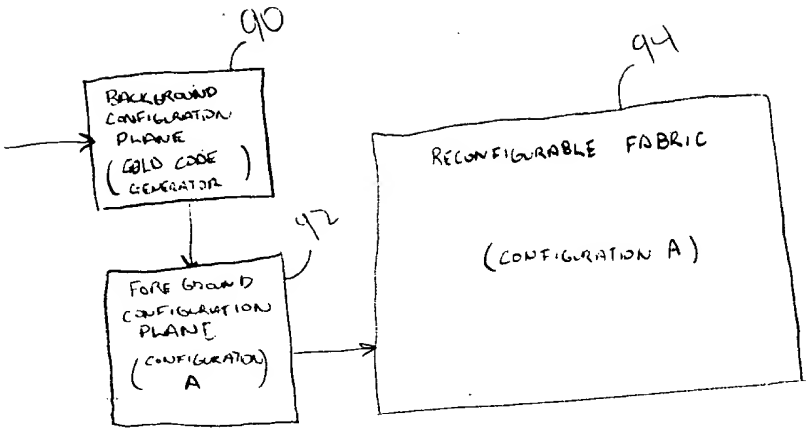


FIGURE 6A

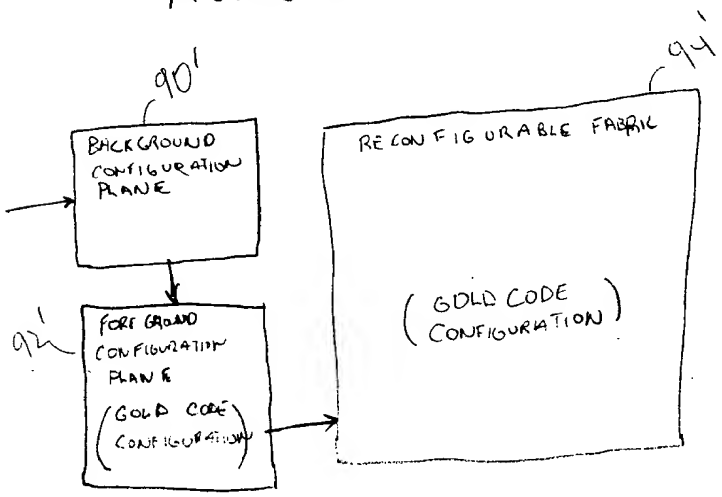


FIGURE 6B

$C_{long1,n}$ = LSFRA[7:0] XOR LSFRB[7:0]
Let us define LFSRC'[i] = LSFRC[2⌊i/2⌋]

$C_{long,n}(i) = C_{long1,n}(i)(1 + j(-1)^i(c_{long2},n(2\lfloor i/2 \rfloor))$ (from 3G TS25.213)

Multiplying bits by +1/-1 is the same as XOR for 0s and 1s.
XORing by 0xAA can be used in place of the $(-1)^i$ term.
In binary representation, the Scrambling Code $C_{long,n}$ becomes:

$C_{long,n}[7:0] = C_{long1,n}[7:0](1 + j(0xAA)$ XOR $C'_{long2,n}[7:0])$

$C_{long,n}[7:0]$ = LFSRA[7:0] XOR LFSRB[7:0]
+j(LFSRA[7:0] XOR LFSRB[7:0] XOR 0xAA XOR LFSRC'[7:0] XOR LFSRD'[7:0]

$C_{long,n}[7:0]$ = SCI[7:0] + jSCQ[7:0]

Let us define LFSRD''[7:0] = 0xAA XOR LFSRD'[7:0], then:
$C_{long,n}[7:0]$ = (LFSRA[7:0] XOR LFSRB[7:0])
+ j(LFSRA[7:0] XOR LFSRB[7:0] XOR LFSRC'[7:0] XOR LFSRD''[7:0])

We use a lookup table to compute LFSRC'[7:0] and LFSRD''[7:0])

FIGURE 7

Gold Code Generator Lookup[6:0] Definitions

At Address 4n+0: OUT[7:0] = Next StateA[3:0]:PASSA[3:0] OUT[7] = IN[6] XOR IN[3] OUT[6] = IN[5] XOR IN[2] OUT[5] = IN[4] XOR IN[1] OUT[4] = IN[3] XOR IN[0] OUT[3] = IN[3] OUT[2] = IN[2] OUT[1] = IN[1] OUT[0] = IN[0]	At Address 4n+2: OUT[7:0] = Next StateC[3:0]:LFSRC'[3:0] OUT[7] = IN[6] XOR IN[3] OUT[6] = IN[5] XOR IN[2] OUT[5] = IN[4] XOR IN[1] OUT[4] = IN[3] XOR IN[0] OUT[3] = IN[2] OUT[2] = IN[2] OUT[1] = IN[0] OUT[0] = IN[0]
At Address 4n+1: OUT[7:0] = Next StateB[3:0]:PASSB[3:0] OUT[7] = IN[6] XOR IN[5] XOR IN[4] XOR IN[3] OUT[6] = IN[5] XOR IN[4] XOR IN[3] XOR IN[2] OUT[5] = IN[4] XOR IN[3] XOR IN[2] XOR IN[1] OUT[4] = IN[3] XOR IN[2] XOR IN[1] XOR IN[0] OUT[3] = IN[3] OUT[2] = IN[2] OUT[1] = IN[1] OUT[0] = IN[0]	At Address 4n+3: OUT[7:0] = Next StateD[3:0]:LFSRD"[3:0] OUT[7] = IN[6] XOR IN[5] XOR IN[4] XOR IN[3] OUT[6] = IN[5] XOR IN[4] XOR IN[3] XOR IN[2] OUT[5] = IN[4] XOR IN[3] XOR IN[2] XOR IN[1] OUT[4] = IN[3] XOR IN[2] XOR IN[1] XOR IN[0] OUT[3] = /IN[2] OUT[2] = IN[2] OUT[1] = /IN[0] OUT[0] = IN[0]

FIGURE 8A

Gold Code Generator Lookup[10:4] Definitions

At Address 4n+0: OUT[7:0] = IN[7:4]:Next StateA[7:4] OUT[7] = IN[3] OUT[6] = IN[2] OUT[5] = IN[1] OUT[4] = IN[0] OUT[3] = IN[6] XOR IN[3] OUT[2] = IN[5] XOR IN[2] OUT[1] = IN[4] XOR IN[1] OUT[0] = IN[3] XOR IN[0]	At Address 4n+2: OUT[7:0] = IN'[7:4]:Next StateC[7:4] OUT[3] = IN[2] OUT[2] = IN[2] OUT[1] = IN[0] OUT[0] = IN[0] OUT[7] = IN[6] XOR IN[3] OUT[6] = IN[5] XOR IN[2] OUT[5] = IN[4] XOR IN[1] OUT[4] = IN[3] XOR IN[0]
At Address 4n+1: OUT[7:0] = IN[7:4]:Next StateB[7:4] OUT[7] = IN[3] OUT[6] = IN[2] OUT[5] = IN[1] OUT[4] = IN[0] OUT[3] = IN[6] XOR IN[5] XOR IN[4] XOR IN[3] OUT[2] = IN[5] XOR IN[4] XOR IN[3] XOR IN[2] OUT[1] = IN[4] XOR IN[3] XOR IN[2] XOR IN[1] OUT[0] = IN[3] XOR IN[2] XOR IN[1] XOR IN[0]	At Address 4n+3: OUT[7:0] = IN''[7:4]:Next StateD[7:4] OUT[3] = /IN[2] OUT[2] = IN[2] OUT[1] = /IN[0] OUT[0] = IN[0] OUT[7] = IN[6] XOR IN[5] XOR IN[4] XOR IN[3] OUT[6] = IN[5] XOR IN[4] XOR IN[3] XOR IN[2] OUT[5] = IN[4] XOR IN[3] XOR IN[2] XOR IN[1] OUT[4] = IN[3] XOR IN[2] XOR IN[1] XOR IN[0]

FIGURE 8B